Predicting Uber Demand in NYC with Wavenet

Long Chen
Urban Big Data Center
Glasgow, UK
Email: long.chen@glasgow.ac.uk

Konstantinos Ampountolas
School of Engineering
University of Glasgow, UK
Email: Konstantinos.Ampountolas@glasgow.ac.uk

Piyushimita (Vonu) Thakuriah
Rutgers University
New Brunswick, NJ, USA
Email: p.thakuriah@rutgers.edu

***Abstract*—Uber demand prediction is at the core of intelligent transportation systems when developing a smart city. However, exploiting uber real time data to facilitate the demand prediction is a thorny problem since user demand usually unevenly distributed over time and space. We develop a Wavenet-based model to predict Uber demand on an hourly basis. In this paper, we present a multi-level Wavenet framework which is a one-dimensional convolutional neural network that includes two sub-networks which encode the source series and decode the predicting series, respectively. The two sub-networks are combined by stacking the decoder on top of the encoder, which in turn, preserves the temporal patterns of the time series. Experiments on large-scale real Uber demand dataset of NYC demonstrate that our model is highly competitive to the existing ones.**

Keywords–Anything; Something; Everything else.

I. INTRODUCTION

With the proliferation of Web 2.0, ride sharing applications, such as Uber, have become a popular way to search nearby sharing rides. Since the Uber demands changes over time at different regions that there is a gap of expectancy between the users and drivers. Various deep learning models have been proposed to facilitate time series data and achieved state-of-the-art performances in a great deal of real-wrold applications. For example, Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM) typically employ a softmax activation function nodes to model correlations of series points. While generally effective, the models of this line, such as [16], largely rely on the hidden state of the previous nodes, which makes it hard to apply the parallel computing tricks within the series. Convolutional Neural Network (CNN) uses trainable receptive field unit to learn local shape patterns. However, most of these time-domain methods fail to consider external information of a time series, although some start to exploit it in a indirect manner.

Inspired by the recent success in a series of time series prediction, Wavenet [30], which employs a convolutional sequence embedding model, has demonstrated its ability of achieving highly competitive performance to the state-of-the-art, i.e., [8], [19]. The logic of Wavenet is to convert the embedding matrix as the "cache" of the previous interactions in a k-dimensional vector and deem the sequential pattens of the interaction as the signature of the "cache". Max pooling operation is then applied to learn the maximum value of the convolutional layer, so as to increase the scope of receptive field and tackle the issues of irregular input length. Fig. 1 depicts the key architecture of Wavenet. To this end, we present a new deep learning method for predicting the Uber demands at an hourly basis, which is a WaveNet-based neural network framework (UberNet) that exploits external features for time series analysis. Simply put, UberNet can utilize both WaveNet-based features (e.g., pickup demands) decomposition and the external features (days of week) to enhance the learning ability of deep neural networks. A max pooling layer is performed to "remember" the maximum value of the hidden layer, in order to magnify the power of receptive field.

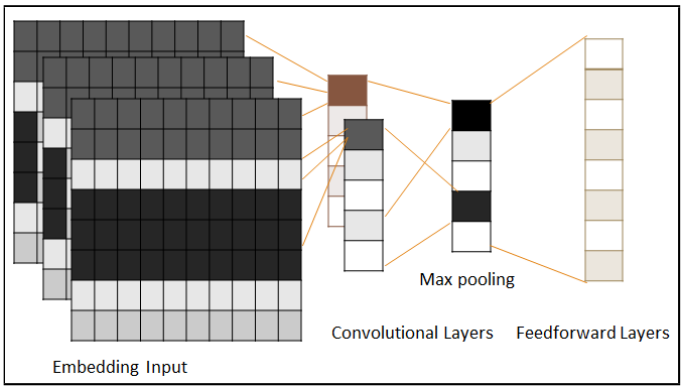

Figure 1. The structure of WaveNet, where different colours in embedding input denote $2 \times k$, $3 \times k$, and $4 \times k$ convolutional filter respectively.

The rest of this paper is organized as follows. We firstly defines the problem of Uber demand prediction with deep learning in Section II. Then, we introduce the related work in Section III. Section IV systematically presents the proposed framework for graph classification. The experimental results of prediction are reported in Section V. Finally, we present our conclusion and future work in Section VI.

II. PRELIMINARY

In this section, Wavenet is shortly recapitulated. Then, we formally introduce several concepts and notations. We introduce a novel neural network that is operated directly on the time series' waveform. The joint probability of a waveform, $x = x1, ..., x_T$, is given as the form of conditional probabilities as follows:

$$P(x) = \prod_{t=1}^{T} p(x_t | x_1, ..., x_{t-1})$$

Each datapoint x_t is therefore conditioned on the value of all previous timesteps. Similar to Wavenet, the core idea

 ISBN: 978-1-61208-692-7

of Ubernet is to embed the previous s timestamp as a $s \times k$ matrix I by taking the embedding query operation, as shown in Fig 1. Each row of the matrix mapped into the latent features of one timestamp. The embedding matrix is the "cache" of the s timestamp in the k dimensional vector. Intuitively, models of various CNNs that are successfully applied in series transformation can be customized to model the "cache" of an traffic demand sequence. However, there are two stark differences, which makes the use of Wavenet model counter-intuitive. First, the variable-length of the demand sequences in real-world entails a large number of "cache" of different sizes, whereas conventional CNN structures with receptive field filters usually fail. Second, the most effective filters for text cannot easily fit into the scenario of modelling demand sequence "caches", since these filters (with respect to row-wise orientation) often fail to learn the representations of full-width embeddings.

Softmax Distribution One approach of learning the conditional distributions $p(x_t|x_1, ..., x_{t-1})$ over the individual timestep is to use a mixture density network. However, van den Oord et al. [2] showed that a softmax distribution gives a superior performance, even if the data is only partially continuous (as is the case for special events or holidays). A possible reason is that a categorical distribution tend to be more flexible which can sift out arbitrary distributions since it has no assumptions about the shape. We adopt the gated activation unit as the gated PixelCNN []

$$z = \tanh(W_{f,k} * x) \odot \mu(W_{g,k} * x) \tag{1}$$

where $*$ is the convolution operator, $\odot$ is an element-wise multiplication, μ represents a sigmoid function, k denotes the layer length, f and g are filter and gate, and W is a convolution filter, which will be detailed in the Section IV. In our preliminary experiments, we found that this non-linearity can greatly outperform the rectified linear activation function for modeling time series.

Given the normalized time series, we propose to use filters in [2], which traverse the full columns of the sequence "cache" by a single large filter. Specifically, the width of filters is usually equal to the width of the input "cache". The height typically varies with respect to the sliding windows over timestamp at a time (Fig 1, embedding input). Filters of different sizes will generate variable-length feature maps after convolution (convolutional layer). To ensure that all maps have the same size, the max pooling is performed over each map, which selects only the largest number of each feature map, leading to a 1×1 map (Fig 1 Feedforward Layers). Lastly, the maps from these filters are concatenated which form a feature vector, which is then fed into a softmax layer that yields the probabilities of next timestamp. In our experiments, we will present results of UberNet with both horizontal and vertical convolutions.

From the above analysis of the convolutions in Wavenet, it is easy to see that there are many drawbacks with the current model. First, the max pooling operator cannot distinguish whether an feature that appear in the map is important or not. In addition, it is oblivious of the position where the max pooling occurs. The max pooling operator may be effective in the "cache" processing, which is actually detrimental for modeling long-range sequences. Furthermore, shallow network of WaveNet will only be suitable for only one hidden layer which often fail when it comes to long-range dependences. The last important drawback derives from the generative process of next timestamp, which we will discuss in Section 5.1.

III. Related Work

Traffic prediction has become a popular topic, where various machine learning approaches have been proposed. In the seminal work of univariate forecasting model, FARIMA is employed to model and predict traffic condition. Min et al. [22] proposed a regression model that considers spatial and temporal interactions of road conditions. However, this model entails a heavy computational cost with a vast amount of parameters, and yet fail to consider several important features of a transportation. Later on, multivariate model of nonparametric regression [6] is designed to exploit the additional features of transportation to infer the traffic state, however, the performance gain was quite limited.

The recent advances in deep learning has provided new opportunities to resolve this problem. Ramdas et al. [28] gives an examination of traditional neural network approaches and they found that the training process of deep learning model is computationally prohibitive when comparing to the traditional ones. To address this issue, researchers have proposed the dropout mechanism [10], which aims to find a sparse representation that is frequently updated in real time. What is more, the approaches of this line still require a fine tuning of parameters, which is not applicable given a different dataset. Lastly, despite the good performance, these models mostly either entails a super-linear execution time [31] or transform the original time series into a constant size format, which may cause a memorization step with unnecessary cost. These problems grow exponentially bad when the length of the series increases.

IV. UberNet

This section introduces our proposed framework there, which uses a novel probabilistic generative model that includes dilated convolutional blocks to create the receptive field. This model is different from the WaveNet models since: (1) the probability estimator can capture the transition process of all individual timestamp at once, instead of the last one in the series; (2) the convolutional layers are on the basis of the more efficient 1D dilated convolution rather than the standard 2D one.

The proposed UberNet framework consists of two core components, a decoder and an encoder. The former is a time series model of one-dimensional layers, which are then masked use dilation. The encoder process the output of decoder into a low-dimensional representation and is defined as a one dimensional convolutional layers as well but without masking. To learn the embedding bandwidth of the encoder and decoder, the decoder is stacked on the computational output of encoder.

Embedding Look-up Layer: Given a time series $t_0, ..., t_l$, the model retrieves each of the first s timestamps $t_0, ..., t_s$ via a look-up operation, and concatenate these timestamp embeddings together. Assuming the embedding dimension is 2k, where k will be set as the number of inner channels in the convolutional network. This results in a matrix of size. Note that unlike WaveNet that considers the input matrix as a 2D

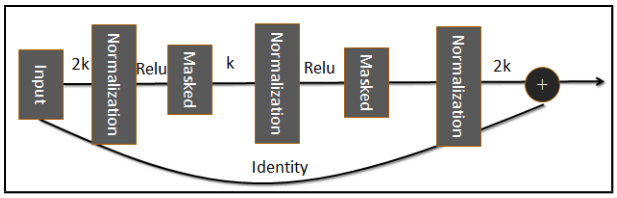

Figure 2. The structure of dilated residual blocks

"cache" during convolution operation, UberNet optimize the embedding via a 1D convolutional operation, which we will detail later.

Dilated layer It is clear in Fig 1 that the standard filter can only operate convolutional function within the receptive field by the depth of the neural network. This makes it difficult to handle long-range sequences. Hence, instead of employing standard convolutions, we designed the dilated convolution to create the generative model, where the dilated layer acts as the convolutional filter to a field which is broader than its original area via the dilation of a zeros matrix. This allow the model become more efficient as it requires fewer parameters. Therefore, a dilated filter is also known as the sparsed filter. Another advantage is that dilated layer does not change the spatial dimensions of the input so that the stacking will be much faster on the convolutional layers and residual structures.

By setting the width of convolutional filter as 3, we can see that the dilated convolutions allow for exponential increase in the size of receptive fields, while the same stacking structure for the standard convolution has only linear receptive fields. The dilated convolutional operation is more powerful to model long-range time series, and thus does not require the use of large filters or additional layers. Practically speaking, one need to carry out the structure of Fig 1 multiple times by stacking in order to further improve the capacity.

Masked Residual Increasing the depth of network layers can generate a higher-level condensed representations, however, it also easily causes some side effects, e.g., the vanishing gradient issue. To address this problem, residual learning mechanism is employed in cnn architectures. While residual learning has been proved to be effective in the field of computer vision,its effectiveness and feasibility in the context of traffic simulation is still largely unknown.

The logic of residual learning is that several convolutional layers can be stacked as a block, from which multiple blocks can communicate with each other through the skip connection scheme by passing signature feature of each block. The skip connection scheme can directly train the residual mapping instead of the conventional identity mapping scheme. This scheme not only maintain the input information but also increased the values of the propagated gradients, resolving the gradient vanishing issue.

Inspired by [24], we employ two residual modules as shown in Fig. 2. We encapsulate each dilated convolutional layer into a residual. The input layer and convolutional one are stacked through a skip connection (i.e., the identity line in Fig 2). Each block is represented as a pipeline structure of several layers, namely, normalization layer, activation (e.g., ReLU) layer, convolutional layer, and a skip connection in a specific manner. In this work we put the state-of-the-art normalization

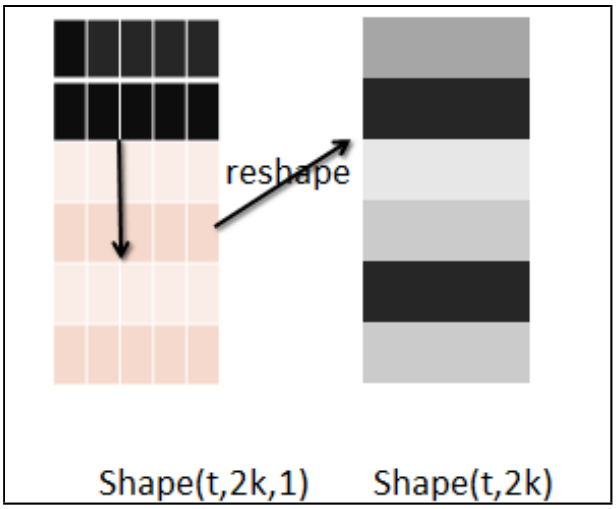

Figure 3. Transformation from the 2D dilated filter to the 1D dilated filter

layer before each activation layer, as it is well known that it has shown superior performance than batch normalization when it comes to sequence processing.

Final Layer The matrix in the last layer of the convolution structure (see Fig 2) has the same size as of the input embedding. However, the output should be a matrix that contains probability distributions of all timestamps in the output series, where the probability distribution is the expected one that generates the prediction results. We use one additional convolutional layer on top of the last convolutional layer in Fig 2. Following the procedure of one-dimensional transformation in Fig 3, we convert the expected prediction matrix as categorical distribution over timestamps, where each row vector is reshaped with the softmax operation.

The aim of this softmax operation is to maximize the log-likelihood of the training data with respect to all hyper-parameters. Fortunately, maximizing log-likelihood can be mathematically converted to minimizing the summation of the binary cross-entropy loss for each timestamp. For practical neural network with tens of millions of timestamps, the negative sampling strategy can be applied to avoid the calculation of the full softmax distributions. The performance operated by these negative sampling strategies is almost the same as the full softmax method given that the sampling size are properly tuned.

V. Experiment Setup

In this section, we will show the statistic of the dataset, the parameter settings, and the experimental results.

A. Datasets

The Uber raw data on pickups is derived from a non-Uber FHV company. The trip information include day of trip, time of trip, pickup location, and driver for-hire license number.

We have chosen datasets which (a) have been made publicly available, so as to enable other researchers to reproduce our results, and (b) have key characteristics covering a large part of the design space (e.g., day of trip, pickup locations). A detailed statistic about this dataset over months is reported in Figure 4. It is clear that the number of pickups is increasing

 ISBN: 978-1-61208-692-7

TABLE I. The performance of pickups prediction with different sets of features(smape)

Learning Models	External Features	Pickup Features	All Features
ARIMA	235.34	243.29	235.22
Linear Regression	324.52	254.02	257.03
Prophet	320.43	**204.12**	**198.72**
LSTM	423.43	351.27	1421.30
UberNet	**203.42**	223.97	227.38

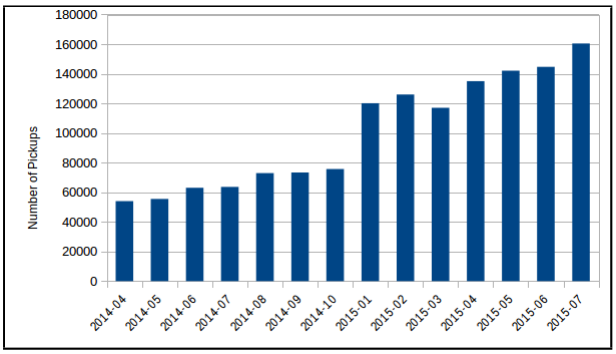

Figure 4. The statistic of pickups over months.

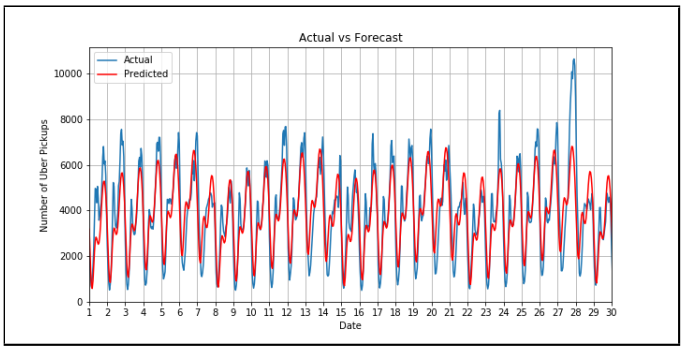

Figure 5. The performance of UberNet during June 2015.

over time. This is due to the fact that more and more people opt into Uber platform over this time period. Notice that we use the 2014 data as the training dataset and the 2015 data as the test dataset.

Approach

In our experiments, we compare the following four approaches:

1) the baseline approach Autoregressive Integrated Moving Average (ARIMA), where the "evolving variable" of interest is regressed on its own lagged (i.e., prior) values and the "regression error" is actually a linear combination of error terms whose values occurred contemporaneously and at various times in the past.
2) the off-the-shelf model Prophet, which is an additive regression model with a piecewise linear kernel. Prophet automatically detects changes in trends by selecting changepoints from the data. The monthly seasonal component is modelled using Fourier series, and a weekly seasonal component is captured using dummy variables.
3) the state-of-the-art Long Short Term Memory (LSTM), which is a type of generalized regression neural networks (GRNN) that aims to relax the constraints of the conventional NN architecture. LSTM used in our work is a simple neural network with four hidden layers (normalization and Relu) with the same number of neurons.
4) the UberNet, which is the approach that described in Section IV

B. Parameter Settings

For ARIMA model, we have the Adjusted Dickey-Fuller Test, where the model achieved the best performance when we set $p = 2$, $d = 1$, and $q = 1$. For Prophet model, we set all the parameters of the default values, the yearly seasonality is set as false. The results of varying models are reported in Table I. A rectifier activation function is used for the neurons in the hidden layer of LSTM. A softmax activation function is used on the output layer to turn the outputs into probability-like values. Logarithmic loss is used as the loss function and the efficient ADAM gradient descent algorithm is used to learn the weights.

Metric Symmetric mean absolute percentage error (SMAPE or sMAPE) is an accuracy measure based on percentage (or relative) errors. It is defined as $smape = \frac{abs(F-A)}{1/2(F+A)}$. This is one of the most commonly used metrics in time series analysis community, e.g., the Web Traffic Time Series Forecasting competition, since this metric is a relative value which ignores outliers and is invariant if the data is linearly rescaled.

C. Experimental Results

Table I shows the performance accuracy of pickup prediction through with different sets of features. It is clear that Prophet and UberNet outperform the other types of learning models. Furthermore, one can see that using both external features, namely, the weather and temperature conditions, works better than using pickup features or relational features alone, for all learning models. Last but not least, one can also observe that RNN approach, i.e., LSTM, fail to yield good results. The reason is probably that the dataset is too small for LSTM to unlock its power. One possible way to remedy this is to carry out hot encoding on all the features so that the feature space can be expanded for learning. We left this for future work.

To further demonstrate the effectiveness of our proposed model, Figure 5 show the performance of *UberNet* during the entire June of 2015. The blue line denotes the ground truth and the red line represent the prediction. It is easy to see that Ubernet can actually predict the pickups most of the time except some holidays. This is probably due to the overfitting issue of the deep learning architecture. UberNet can largely

resolve this issue but cannot eradicate it. We believe UberNet will achieve a even better result given more training data.

VI. Conclusion

The major contribution of this paper is threefold. First, we formally defined the problem of Uber pickup prediction as a machine learning task. Second, we identified several external features which can be used together with standard pickup features by learning algorithms to predict pickups in terms of their context. Third, we demonstrate that it is better to exploit both external features and pickup features through a WaveNet type deep neural network, namely, UberNet.

There are several interesting and promising directions in which this work could be extended. First, since users and Uber drivers can be represented as a graph structure, it will be interesting to learn the performance of some advanced graph regression algorithms, such as BB-Graph [2]. Second, UberNet in the current form relies on one of the simplest convolutional neural netowrk, which makes sense as a first step towards integrating the context information of Uber into learning model, but of course we could consider using more sophisticated neural network like RCNN. Lastly, For example, location of the user has not considered in this work yet , in future we would like to explore the geographical distribution to facilitate UberNet.

References

[1] Osvaldo Anacleto, Catriona Queen, and Casper J Albers. Multivariate forecasting of road traffic flows in the presence of heteroscedasticity and measurement errors. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 62(2):251–270, 2013.

[2] Merve Asiler and Adnan Yazıcı. Bb-graph: A new subgraph isomorphism algorithm for efficiently querying big graph databases. *arXiv preprint arXiv:1706.06654*, 2017.

[3] Aharon Bar-Hillel, Tomer Hertz, Noam Shental, and Daphna Weinshall. Learning a mahalanobis metric from equivalence constraints. *Journal of Machine Learning Research*, 6(Jun):937–965, 2005.

[4] C Borgelt and M Berthold. Finding relevant substructures of molecules: Mining molecular fragments. In *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002)*, pages 51–58, 2002.

[5] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[6] Stephen Clark. Traffic prediction using multivariate nonparametric regression. *Journal of transportation engineering*, 129(2):161–168, 2003.

[7] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pages 3844–3852, 2016.

[8] Felix A Gers, Douglas Eck, and Jürgen Schmidhuber. Applying lstm to time series predictable through time-window approaches. In *Neural Nets WIRN Vietri-01*, pages 193–200. 2002.

[9] Huahai He and Ambuj K Singh. Query language and access methods for graph databases. In *Managing and mining graph data*, pages 125–160. Springer, 2010.

[10] David P Helmbold and Philip M Long. On the inductive bias of dropout. *The Journal of Machine Learning Research*, 16(1):3403–3454, 2015.

[11] Jun Huan, Wei Wang, and Jan Prins. Efficient mining of frequent subgraphs in the presence of isomorphism. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 549–552. IEEE, 2003.

[12] Akihiro Inokuchi, Takashi Washio, and Hiroshi Motoda. An apriori-based algorithm for mining frequent substructures from graph data. *Principles of Data Mining and Knowledge Discovery*, pages 13–23, 2000.

[13] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[14] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[15] George Kuramochi, Michihiro andKarypis. Frequent subgraph discovery. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 313–320. IEEE, 2001.

[16] Nikolay Laptev, Jason Yosinski, Li Erran Li, and Slawek Smyl. Time-series extreme event forecasting with neural networks at uber. In *International Conference on Machine Learning*, number 34, pages 1–5, 2017.

[17] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.

[18] Lei Lin, Zhengbing He, and Srinivas Peeta. Predicting station-level hourly demand in a large-scale bike-sharing network: A graph convolutional neural network approach. *Transportation Research Part C: Emerging Technologies*, 97:258–276, 2018.

[19] Zachary C Lipton, David C Kale, Charles Elkan, and Randall Wetzel. Learning to diagnose with lstm recurrent neural networks. *arXiv preprint arXiv:1511.03677*, 2015.

[20] Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, Fei-Yue Wang, et al. Traffic flow prediction with big data: A deep learning approach. *IEEE Trans. Intelligent Transportation Systems*, 16(2):865–873, 2015.

[21] Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, et al. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

[22] Wanli Min and Laura Wynter. Real-time road traffic prediction with spatio-temporal correlations. *Transportation Research Part C: Emerging Technologies*, 19(4):606–616, 2011.

[23] Jennifer Neville and David Jensen. Iterative classification in relational data. In *Proc. AAAI-2000 Workshop on Learning Statistical Models from Relational Data*, pages 13–20, 2000.

[24] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *International Conference on Machine Learning*, pages 2014–2023, 2016.

[25] Siegfried Nijssen and Joost N Kok. A quickstart in frequent structure mining can make a difference. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 647–652. ACM, 2004.

[26] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM, 2014.

[27] Nicholas Polson, Vadim Sokolov, et al. Bayesian analysis of traffic flow on interstate i-55: The lwr model. *The Annals of Applied Statistics*, 9(4):1864–1888, 2015.

[28] Aaditya Ramdas and Ryan J Tibshirani. Fast and flexible admm algorithms for trend filtering. *Journal of Computational and Graphical Statistics*, 25(3):839–858, 2016.

[29] John Rice and Erik Van Zwet. A simple and effective method for predicting travel times on freeways. *IEEE Transactions on Intelligent Transportation Systems*, 5(3):200–207, 2004.

[30] Aäron Van Den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. In *SSW*, page 125, 2016.

[31] Junbo Zhang, Yu Zheng, and Dekang Qi. Deep spatio-temporal residual networks for citywide crowd flows prediction. In *AAAI*, pages 1655–1661, 2017.

 ISBN: 978-1-61208-692-7